Description of All Components

~~WelcomeView~~

- ~~*This component is a view that displays the vibrant welcome screen that introduces the player to the game's title and allows them to open the world map.*~~
- ~~*Only the **WorldMapView** communicates with the model. It communicates the following:*~~
  - ~~*The **WelcomeView** communicates to the **WorldMapView** to navigate to it*~~

~~WorldMapView~~

- ~~*This component is a view that displays the world map that has the Math, Geography, and Chemistry islands.*~~
- ~~*The **DifficultyView** and **GameView** communicate with the model. They communicate the following:*~~
  - ~~*The **WorldMapView** communicates to the **DifficultyView** to navigate to it*~~
  - ~~*The **WorldMapView** communicates with the **GameView** to update which question type is selected*~~

~~DifficultyView~~

- ~~*This component is a view that displays the difficulty selection screen, which changes the map to one that has an enemy path length that is either easy, medium, hard, or EXTREME*~~
- ~~*The **WorldMapView** and **GameView** communicate with the model. It communicates the following:*~~
  - ~~*The **WorldMapView** communicates to the **DifficultyView** to navigate to it*~~
  - ~~*The **Difficulty** communicates to the **GameView** to navigate to it and pick which type of map is selected*~~

GameView

- *This component is a view that displays the main GUI for the game once a user is in level. It has the grid map in the top middle, a question screen on the left side, a menu at the bottom middle, and a tower purchase screen on the right side.*
- *Several different components communicate with the **GameView.** The following is communicated:*
  - *The **UserController** updates components in the viewer, while the **GameView** communicates user actions to the Controllers*
  - *The **MapModel** supplies the map data to GameView so it can construct the maps correctly*

~~***UserController:***~~

- ~~*These components are controllers that perform functions for the objects and update the components in the viewers.*~~
- ~~*Several different components communicate with these Controllers. The following is communicated:,*~~
    - ~~*The **GameView** takes input and viewed data and passes it to the Controllers to perform logic*~~ *user controller and model have been determined to be potentially redundant when compared to the functionality being implemented to GameView, may still be implemented with this structure but possibly not.*

MapModel:

- *This component contains the physical construction of the 4 maps within a 2d array of tile objects*
- *Tiles can either be land, water, border, or enemyPath tiles*
- *GameView accesses the array of Tile locations and uses it to understand what textures and functionality needs to be placed in the GUI*

TowerController, and EnemyController:

- *These components are the controllers of the abstract classes of the towers and enemies*
- *Several different components communicate with these controllers. The following is communicated:*
    - *The **GameView** takes input and viewed data and passes it to the controllers to perform logic. The user interacts with the **GameView** which passes information on to the controller to change the respective instance classes.*

TowerProperties, Enemy Model:

- *These components are abstract classes that serve as the frameworks for specific towers and enemies*
- *Several different components communicate with these models. The following is communicated:*
    - *The models communicate data to the Controllers that will be processed for game logic*
    - *The models communicate with the child classes which inherit methods from the abstract parent classes*

Enemy examples:

- *A beetle that has more health than the standard bug enemy*
- *A praying mantis boss that appears at the end of a level*

Tower examples:

- *A boat tower that is the only type of tower that can be placed in water but is more expensive*
- *A lightning tower that has chain damage that hits multiple enemies at once*

TowerModelI (Interface):

- *This interface is used for the firing strategy method of the towers. It only has the abstract method Fire which is implemented in all of the tower classes.*
- *The interface communicates with all of the towers and requires them to to individually create their fire strategy method.*

Wave:

- *This class creates the wave objects that come through the map.*
- *The class communicates with all of the towers in their fire method:*
    - *The getWave method returns each individual wave as an arrayList for the towers to access the enemies inside and shoot at them.*

UML link:
UML DIAGRAM @ LUCIDCHART

[Brand new diagram boxes for v2 are in green, additionally many v1 descriptions were added to reflect the actual return types and method/attribute names that are currently in use].

Edit link:
https://lucid.app/lucidchart/bc909f64-719f-44c7-8e64-4526d0dcfa37/edit?viewport_loc=419%2C-608%2C2159%2C2415%2C0_0&invitationId=inv_ec6e4aba-9cc9-49e5-84ab-c69256d90cb3

Entity Diagram:

- **Entities:**
    - Category (represents the categories of questions like Math, Geography, Chemistry).
    - Question (represents the questions and answers).
- **Relationships:**
    - A Category has many Questions.
    - Each Question belongs to a single Category.

Diagram Elements:

- **Category:**
    - id: Unique identifier for the category (Primary Key).
    - name: Name of the category (e.g., Math, Chemistry).
- **Question:**
    - id: Unique identifier for the question (Primary Key).
    - question_text: The text of the question.
    - answer: The answer to the question.
    - category_id: Foreign key linking to the Category table.

ID:	**Name:**
1	Math
2	Geography
3	Chemistry

ID	**question_text**	**answer**	**category_id**
1	What is 1 x 1	1	1

2	What is capital of Colorado	Denver	2
3	What is symbol for Hydrogen	H	3